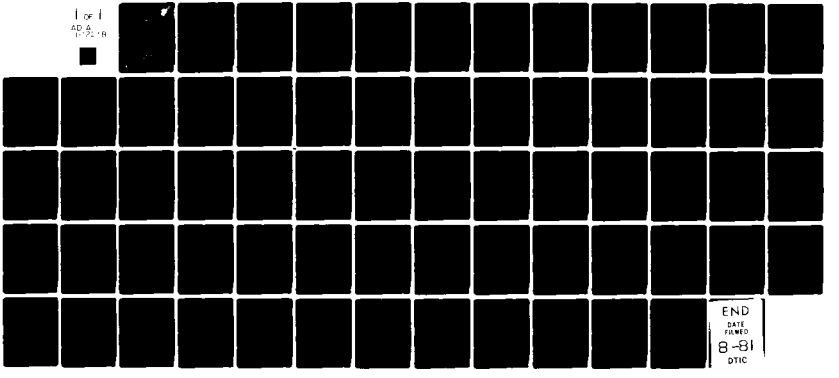
AD-A102 218 GENERAL RESEARCH CORP SANTA BARBARA CA F/G 9/2
FAVS ENHANCEMENTS.(U)
MAY 81 R A MELTON F30602-79-C-0156
UNCLASSIFIED RADC-TR-81-94 NL



RADC-TR-81-94
Final Technical Report
May 1981

FAVS ENHANCEMENTS

General Research Corporation

R.A. Melton

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-94 has been reviewed and is approved for publication.

APPROVED: Frank S LaMonica

FRANK S. LAMONICA
Project Engineer

APPROVED: Alan R Barnum

ALAN R. BARNUM, Assistant Chief
Information Sciences Division

FOR THE COMMANDER: John P Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISIE) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-81-94	2. GOVT ACCESSION NO.: AD-A102218
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): FAVS ENHANCEMENTS .	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report May 79 - June 80
	6. PERFORMING ORG. REPORT NUMBER: N/A
7. AUTHOR(s): R.A. Melton	8. CONTRACT OR GRANT NUMBER(s): F30602-79-C-0156
9. PERFORMING ORGANIZATION NAME AND ADDRESS: General Research Corporation, PO Box 6770, Santa Barbara CA 93111	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 64701B, 43050307
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISIE), Griffiss AFB NY 13441	12. REPORT DATE: May 1981
	13. NUMBER OF PAGES: 72
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Frank S. LaMonica (ISIE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Software Testing
Computer Software Verification
Software Development Tool
FORTRAN

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report documents the results of an effort to upgrade the FORTRAN Automated Verification System (FAVS) installations at the Defense Mapping Agency (DMA) Aerospace Center (DMAAC) and Hydrographic/Topographic Center (DMAHTC). The upgrade, which was accomplished by a series of four software releases, included efficiency enhancements, user interface improvements, and the capability to process the ASCII FORTRAN programming language.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

402754

CONTENTS

FIGURES

TABLES

ACKNOWLEDGEMENTS

Many individuals contributed to the design and implementation of FAVS. E. F. Miller, Jr., originated the work at General Research Corporation which resulted in the methodology for FAVS. He, together with Michael Paige, Jeoff Benson, Randy Urban, Rich Melton, Carolyn Gannon, Dick Wisehart, and others, initially built RXVP, an automated verification system for FORTRAN, a product of the Program Validation Project sponsored by General Research Corporation.

JAVS (JOVIAL Automated Verification System) was the immediate successor to RXVP. In its development, the RXVP software testing methods were examined, the algorithms were extended to JOVIAL constructs, and the JAVS software itself was written in the JOVIAL language. JAVS was installed at RADC, and user training was conducted. The major contributors to the JAVS project were Jeoff Benson, Nancy Brooks, Carolyn Gannon, E. F. Miller, Jr., Ray Stone, Randy Urban, and Dick Wisehart, all employees (at that time) of General Research Corporation. The RADC Project Engineer for JAVS was Dick Robinson.

STRUCTRAN-1 and STRUCTRAN-2 were developed concurrently with the JAVS project. STRUCTRAN-1 translates DMATRAN (a structured extension of FORTRAN) to FORTRAN, and STRUCTRAN-2 translates FORTRAN to DMATRAN. The STRUCTRAN software was installed at DMAAC in St. Louis, Missouri. The major contributors to STRUCTRAN were Dorothy Andrews, Rich Melton, and Randy Urban. The RADC Project Engineer for STRUCTRAN was Don Mark.

The development of FAVS integrated concepts from RXVP with STRUCTRAN-2, incorporated certain capabilities of JAVS, extended the STRUCTRAN-2 capabilities, and improved STRUCTRAN-1. The FAVS software is written in DMATRAN. FAVS has been installed at DMAAC, DMAHTC, and RADC, and user training and maintenance training have been conducted.

The major contributors to FAVS were Dorothy Andrews, Carolyn Gannon, Rich Melton, and Randy Urban. The RADC Project Engineer was Frank LaMonica.

The FAVS enhancements described in this report reduced the computer resources required during FAVS processing by a factor of three or more, improved the FAVS user interface and reports, and added processing of the UNIVAC ASCII FORTRAN dialect. The enhanced FAVS has been installed at DMAAC and DMAHTC. The major contributors to FAVS enhancements were Dorothy Andrews and Rich Melton. The RADC Project Engineer was Frank LaMonica.

EVALUATION

The purpose of this contractual effort was to upgrade the FORTRAN Automated Verification System (FAVS) installations at the Defnese Mapping Agency (DMA) Aerospace Center (DMAAC) and Hydrographic/Topographic Center (DMAHTC). The goals of the effort were to provide efficiency enhancements to reduce FAVS processing time by at least a factor of three, improve the FAVS user interface, extend FAVS processing to UNIVAC ASCII FORTRAN, reduce FAVS data base library storage requirements, and to correct residual errors. The upgrade, which was accomplished by a series of four software releases, was successfully performed and, in some cases, the goals were exceeded. The effort resulted in a software testing tool which is closely tuned to the DMA processing environment. This effort was responsive to the objective of the RADC Technology Plan, TPO 4G3, "Software Engineering (Software Tools)."

Frank S LaMonica

FRANK S. LAMONICA
Project Engineer

1 INTRODUCTION

The FORTRAN Automated Verification System (FAVS) is a series of tools which provide:

- Translation from DMATRAN (a structured extension of FORTRAN) to FORTRAN and from FORTRAN to DMATRAN
- Static detection of unreachable statements, set/use errors, mode-conversion errors, and external reference errors
- A means of measuring the effectiveness of software test cases, both individually and cumulatively
- Assistance in the construction of test data that will thoroughly exercise the software
- Automated documentation

As part of its program for applying advanced technology to improve the quality and reliability of software, and to provide testing tools for the Defense Mapping Agency, Rome Air Development Center contracted with General Research Corporation to design, install, and document certain enhancements of FAVS. The enhancements are intended to reduce the cost of assuring that software systems written in FORTRAN are comprehensively tested. This report (the final report for the project) describes the enhancements and quantifies the improvement in performance where possible.

The work involved the application of efficient and automatable algorithms and techniques to the verification and testing of FORTRAN software. The specific tasks were to provide efficiency enhancements that reduce FAVS processing time by at least a factor of three, improve the FAVS user interface and reports, extend FAVS processing to UNIVAC ASCII FORTRAN, provide an interface library capability for FAVS, and correct errors in FAVS.

The FAVS software was enhanced in a series of four updates (Table 1.1). Each update was installed and tested at DMAAC and DMAHTC by GRC personnel, after verifying correct operation by testing on the CDC 6400 at GRC in Santa Barbara, California. It was originally planned that all efficiency enhancements would be completed during Update 1. At that time the CDC 6400 version of FAVS had met the goal of running in one-third of the original processing time, mostly because of reductions in CPU time. When Update 1 was installed on the UNIVAC 1100/80s at the DMA sites,however, the relative improvement was not as great. The slower I/O devices in the UNIVACs caused FAVS to be more I/O bound. Subsequent updates, therefore, included further enhancements to improve the I/O performance of FAVS. The result was significant further improvement in performance on the CDC 6400 as well as the UNIVAC 1100/80.

TABLE 1.1

FAVS UPDATES

FAVS Update	Week of Update DMAAC	DMAHTC	Type of Update
1	7-23-79	7-16-79	Efficiency enhancements, error corrections
2	10-15-79	11-11-79	Efficiency enhancements, report enhancements, error corrections
3	1-21-80	1-28-80	ASCII FORTRAN, error corrections, efficiency enhancements
4	5-18-80	4-21-80	Interface library, user interface enhancements, error corrections

The enhanced FAVS has been implemented for the analysis of computer software written in UNIVAC FORTRAN V, ASCII FORTRAN, or DMATRAN and is operational on the UNIVAC 1100/80 computers at DMAHTC in

Washington, DC, and DMAAC in St. Louis, Missouri, and on the CDC 6400 computer at General Research Corporation in Santa Barbara, California, where it was developed.

Section 2 of this report summarizes the efficiency enhancements. Section 3 describes restart file enhancements. Section 4 describes extensions for the UNIVAC ASCII FORTRAN dialect. Section 5 describes the enhanced FAVS user interface and improved FAVS reports. Section 6 describes error and deficiency corrections to FAVS.

In addition to this report, a number of other reports have been prepared as part of the effort:

- FAVS (FORTRAN Automated Verification System) User's Manual (CR-1-754/1, December 1977, revised April 1980).
  This report is an introduction to using FAVS in the testing process. Its purpose is to acquaint the user with the application of FAVS to program testing, so that an efficient approach to program verification can be taken. The basic commands by which FAVS provides this assistance are discussed in detail. FAVS processing is described in the order normally followed by the beginning FAVS user. The Appendixes include a description of FAVS operation at DMAHTC and DMAAC, with both sample command sets and sample job control statements.

- DMATRAN User's Guide (CR-1-673/1, January 1978)
  This report describes the structured constructs and syntax of DMATRAN, a structured extension to FORTRAN. It also details the use of the DMATRAN preprocessor, which translates DMATRAN into FORTRAN. Procedures for using the UNIVAC 1100/80 or the Honeywell 6180 version of DMATRAN are included. Revision pages to update the DMATRAN User's Guide are included as Appendix A of this Final Report.

- FAVS (FORTRAN Automated Verification System) Computer Program Documentation: Vols. 1, 2, 3 (CR-2-754/1, January 1978; Revised January 1981).

  These reports describe the FAVS software design, the organization and contents of the FAVS data base, and for each FAVS component its function, each of its invokable modules, and the global data structures it uses. The report is intended for use in FAVS software maintenance, together with the Software Analysis reports described below.

- FAVS Computer Program Documentation: Vol. 4, Software Analysis

  This volume is a collection of computer output produced by FAVS, not reproduced but on file at RADC, DMAAC, and DMAHTC. The source code for each component of the FAVS software has been analyzed by FAVS itself to produce enhanced source code listings of FAVS with indentation and control structure identification, inter-module dependence, all module invocations, module control structure, and a cross reference of symbol usage. This volume is intended to be used with Vols. 1-3 for FAVS software maintenance. It is itself also an excellent example of the use of FAVS for computer software documentation.

2 EFFICIENCY ENHANCEMENTS

Experience at DMA indicated that FAVS required excessive computer time when analyzing large FORTRAN systems. The time (especially for I/O processing) seemed to increase exponentially as more lines of source code were analyzed. The DMA data processing department was running large backlogs--FAVS users were faced with overnight (or even weekly) turnaround--and DMA management could foresee even larger backlogs if the use of FAVS became widespread. To remedy this situation, a goal was set of enhancing FAVS efficiency to reduce its processing time (CPU and I/O combined) by a factor of three. This goal was met and exceeded. FAVS processing time (especially I/O processing time) is now nearly linear in source lines analyzed. In processing 3500 to 7000 lines of source code, processing time has been reduced by a factor of 8 to 10.

Figures 2.1 and 2.2 indicate performance increases in the range of 150 to 7000 source lines. In these figures FAVS.0 refers to FAVS before any efficiency enhancements, FAVS.4 refers to FAVS after the enhancements described in this section. Table 2.1 shows characteristics of the source programs used, as well as the actual processing seconds used (on the CDC 6400/GOLETA system). All source programs used were written in DMATRAN, so that compilation time refers to the sum of DMATRAN precompilation and CDC SCOPE FTN compilation time. All FAVS times are for the STATIC option. Similar results were obtained on the DMA UNIVAC 1100/80.

I/O Processing Time

As indicated in Fig. 2.1, FAVS I/O processing time was improved by a factor of 3 to 7 when processing 150 source lines, but was improved dramatically (10 to 30 times) when processing 3500 to 7500 source lines. Previously, FAVS.0 I/O processing time increased exponentially with the number of symbols processed. FAVS.4 I/O processing is nearly a constant

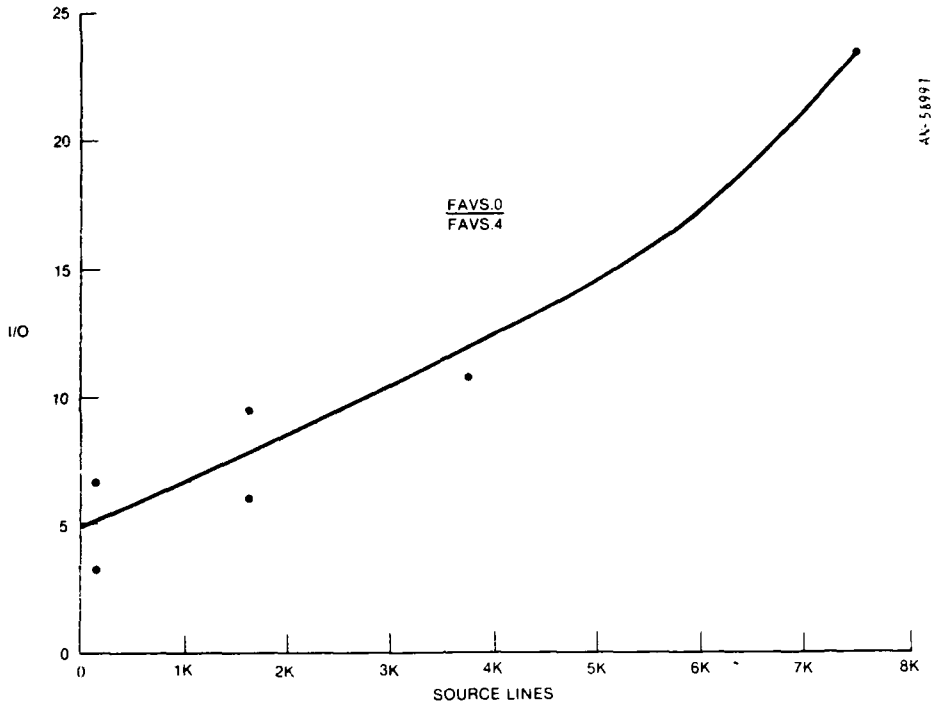


Figure 2.1. Relative Improvement in FAVS I/O Processing Time

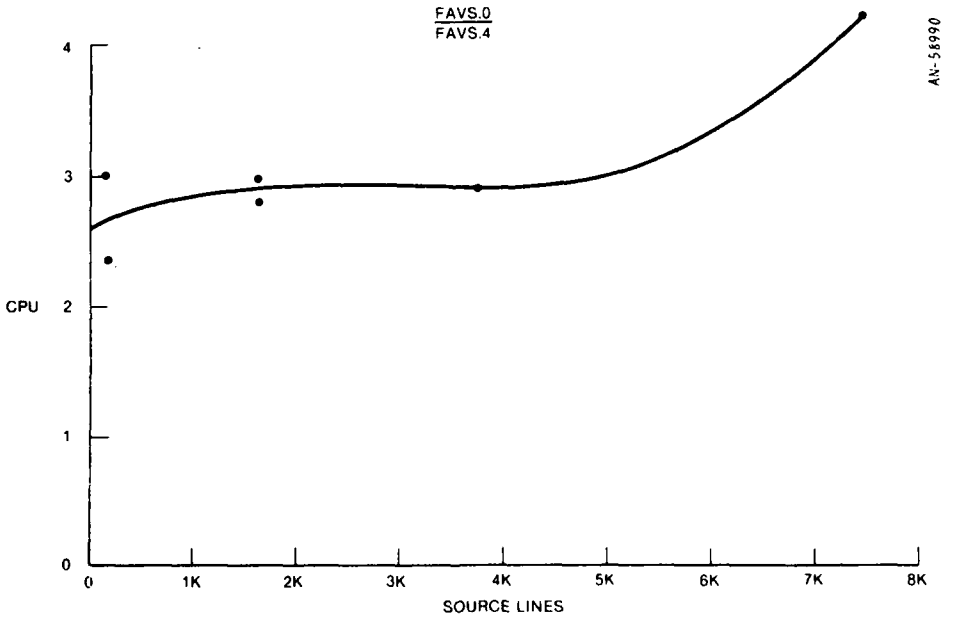


Figure 2.2. Relative Improvement in FAVS CPU Processing Time

TABLE 2.1

CHARACTERISTICS OF SOURCE PROGRAMS AND CDC 6400 PROCESSING TIMES (IN SECONDS)

NUMBER OF MODULES	NUMBER OF LINES	FAVS.0 CPU	FAVS.4 CPU	FAVS.0 I/O	FAVS.4 I/O	FAVS.0 RESTART	FAVS.4 RESTART
6	154	27.1	8.9	86	12.7	26.5K	6K
1	188	39	16.0	45	13.8	17K	6K
2	1612	437	148.1	710	75.0	74.5K	9K
6	1546	372	133.0	468	78.0	81K	8.5K
44	3751	886	305	1845	174	261.5K	12.5K
69	7400	2780	656	8200	355	502K	31K

multiple of source lines processed. FAVS.4 performs extremely well in terms of I/O time as more and more symbols are processed, primarily because of the enhancements in token processing and cross-reference processing. FAVS.0 exhibited a high I/O-processing overhead when processing small numbers of source lines and symbols and large numbers of modules. In FAVS.4, this problem was eliminated by enhancements to the virtual data space processing.

CPU Processing Time

As indicated in Fig. 2.2, FAVS CPU processing time was improved by a factor of two to three when processing 150 source lines and by a factor of as much as four when processing 7500 source lines. FAVS CPU processing time appears to be a constant multiple of source lines processed. This combined with recent improvements in the DMA computers translates into an order of magnitude reduction in CPU processing time in the DMA data processing environment. At the same time as the FAVS Enhancement contract, DMA undertook a series of hardware enhancements which reduce CPU processing time by a factor of 3 to 4. FAVS users at DMA can now expect significantly improved turnaround, and DMA managers are not faced with the prospect of significantly increasing backlogs as FAVS becomes more widely used.

2.1 TOKEN PROCESSING

Meaningful strings of source text characters (keywords, labels, symbols, constants, etc.)--which may be longer than one computer word--are internally represented by FAVS as "token integers". The way in which tokens were processed contributed to the nonlinear behavior of FAVS when processing source elements containing many symbols. Token integers were implemented as pointers to character strings stored in a "token table". When FAVS encountered a character string, it performed a sequential search of all entries in the token table to see if the string had been previously encountered; if not, it added the string as the last entry of the token table.

FAVS token processing was modified to recognize short character strings which fit in one machine word (6 characters or less on the UNIVAC 1100/80) and treat the string (interpreted as an integer) as its own token integer, in the following way. FAVS character strings are left-justified and blank-filled. Short character strings are now converted into token integers by filling them with a special encoding character (thus storing the character string as well as its length into one machine word). There is no need to check the table at all for short strings. Long character strings, and character strings ending in the special encoding character (used for filling short character strings), are treated in the same manner as before. Since most FORTRAN tokens are short (operators, delimiters, and symbols), this change significantly reduces the size of the token table as well as the number of times it is searched. Encoding short character strings into token integers and decoding token integers into character strings is an efficient process which is independent of the number of tokens previously processed.

2.2 CROSS-REFERENCE PROCESSING

Several of the FAVS options require a data structure which identifies the occurrence and use of all symbols in a set of source elements. This was previously implemented as a Referenced Symbol table which contained an entry for each symbol, and a Cross Reference table which contained an entry for each use of each symbol. The Referenced Symbol entry started a chain of entries in the Cross Reference data structure for all occurrences of that symbol. The global scope of the Referenced Symbol table contributed to the nonlinear behavior of FAVS when processing source elements containing many symbols. Previously for each use of a symbol in any source element, a linear search of the Referenced Symbol table was performed to see if a chain of uses for the symbol already existed. If it did, this use was added to the chain; otherwise a new chain was begun.

FAVS was modified to include in each module's symbol table a pointer to the use chain of each symbol used in the module. This improves the efficiency of FAVS cross-reference processing by completely eliminating the need to search the Referenced Symbol table. Because it also directly relates cross-reference information to symbol information, better cross-reference and static analysis reports can be produced.

2.3 SYMBOL TABLE PROCESSING

Each symbol, label, or constant which occurs in a source element was previously described by a 19-word entry in the element's symbol table. The size of this data structure contributed significantly to the size of a FAVS restart file, as well as to the processing overhead. The symbol table was modified to utilize 7-word entries plus expandable variable-length entries. These modifications reduce the size of FAVS restart files and decrease FAVS processing time.

2.4 SOURCE TEXT ANALYSIS

FAVS previously performed three passes through the original source text before converting it to an internal form. This required reading the source three times and writing it twice, with minor alterations each time it was written. FAVS was modified to perform the same function while reading the original source only once.

FAVS previously translated source text to an internal form, reconstructed the source text from that internal form, and then stored the reconstructed text for use in printing and punching. FAVS was modified to store source text exactly as it is read in and to tie its internal statement representations to the original source text. This eliminates a time-consuming character processing task (source text reconstruction) as well as improving the usability of FAVS reports.

2.5 DATA BASE SYSTEM TUNING

The above enhancements concentrated on the problem of analyzing FORTRAN source code more efficiently. FAVS uses a general purpose data base storage and retrieval system. After the above enhancements were completed, it was determined that the data base system accounted for 95% of FAVS I/O time. By tuning the data base system, FAVS I/O was reduced by a factor of four (in addition to the reductions described earlier).

The data base system uses a paging technique in which the least frequently used "page" is written out of core when a page not in core is called for. A page fault occurs when the desired page is not core resident. A data structure may occupy more than one page; page switching occurs when access to the same data structure crosses a page boundary. Tuning of the data base system consisted of the following changes.

1. Previously the initial reference to a page was treated as a page fault (a page of zeroes was written onto mass storage and then read into core). This artifically increased the page fault rate. Now an appropriate in-core page is initialized to zeroes.

2. Previously a page could only contain information about one module. This led to high fragmentation and a higher page fault rate. Now a page can contain information about more than one module.

3. Previously the "activity status" algorithm that kept track of the frequency of use of in-core pages made note of each page fault or page switch. It ignored accesses to data structures completely contained in one page. A higher page fault rate resulted, since frequently referenced pages were likely to have been "paged out". Now an equivalent but computationally simpler activity status algorithm keeps track of each access to a page, substantially reducing the page fault rate.

4. Previously 20 in-core pages of 500 words provided optimum performance for a wide range of test points. Now 30 in-core pages of 300 words significantly reduces the page fault rate for the same range of test points.

5. The minimum number of in-core pages required for internal data base pointers (as opposed to source text data structures) was reduced from 4 to 2. Previously 20% of the in-core area was used up by data base pointers; now it is only 5.4%. This also helps to reduce the page fault rate.

3 RESTART FILE ENHANCEMENTS

FAVS is designed to be especially useful for analyzing large FORTRAN systems consisting of many compilation units. Section 2 described the enhancements used to reduce FAVS processing time for large FORTRAN systems. This section describes changes to the FAVS restart file (permanent data base) which have produced an additional order of magnitude reduction in the cost of consistently using FAVS during the coding, test and acceptance, and maintenance of software. This reduction is achieved by:

1. Reducing the size (and storage costs) of FAVS restart files by an order of magnitude or more.
2. Producing updated versions of FAVS global reports by reanalyzing changed modules only.

In addition, the limitation of 100 compilation units per restart file was completely removed.

Restart processing time using FAVS.0 was a function of the number of source lines previously analyzed as well as the number of new or modified source lines to be analyzed. The size of the restart file--10 to 20 times the size of the source text--effectively limited the use of restart files to small systems. Besides, it was observed that reprocessing a complete system was not much more expensive than reanalyzing only changed modules using a restart file. As a result, FAVS.0 was used almost entirely in its stand-alone mode. With FAVS.4, however, the enhancements in efficiency and in the restart file make it economically feasible to use the FAVS restart mode throughout the coding, test and acceptance, and maintenance of large FORTRAN software systems. This makes it possible to constantly verify that interfaces between modules are correct and consistent, and to produce updated system level documentation as required.

Fig. 3.1 and Fig. 3.2 indicate that FAVS processing time using a restart file has been dramatically improved, especially for large systems with many source lines.

Restart File Size

As indicated in Fig. 3.2, the size of the FAVS restart file was reduced by a factor of five to fifteen times. This reduction was accomplished by a major change in the information stored on the restart file. Previously, the file contained detailed descriptions of source text, which could be reanalyzed (but not modified) without having to read source text from an external file, and used to produce global information and reports.

The new FAVS restart file saves the information necessary to produce reports about more than one module and perform static analysis on new or modified source text, but it does not save source text or a detailed description of it. To reanalyze given source text, the text must be read from an external file. This change is consistent with the improved processing efficiency of FAVS.4, which makes it cheaper to reprocess source text than to store detailed source text descriptions for possible later use.

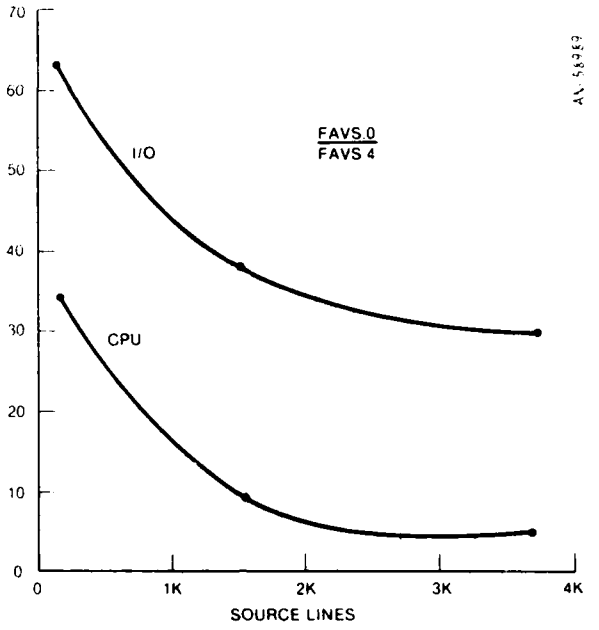


Figure 3.1. Relative Improvement in FAVS Restart Processing

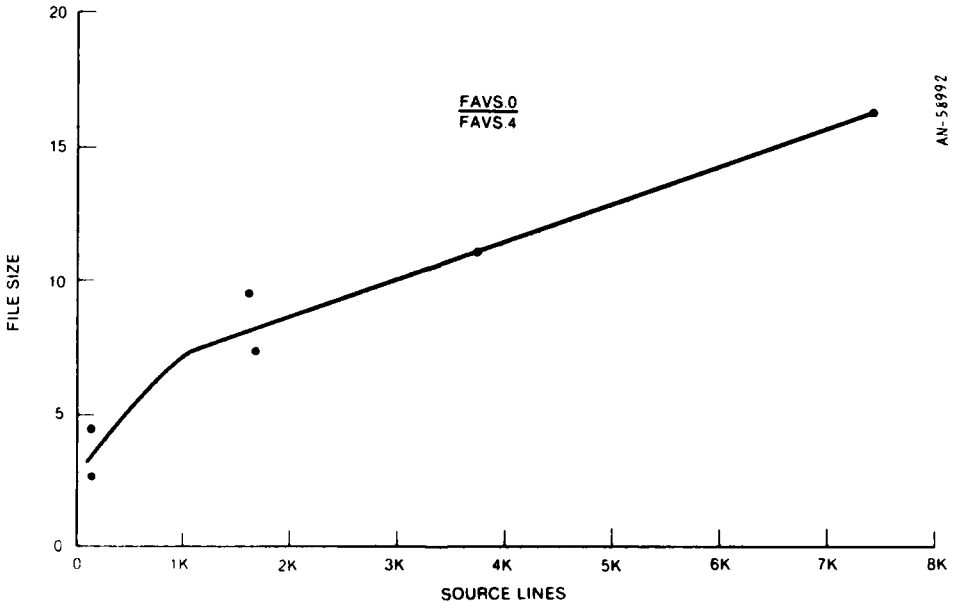


Figure 3.2. Relative Improvement in Restart File Size

4 UNIVAC ASCII FORTRAN ENHANCEMENTS

In order to enhance FAVS and DMATRAN as operational tools for ASCII FORTRAN (UNIVAC's version of FORTRAN 77), GRC modified FAVS to recognize and process ASCII FORTRAN and modified DMATRAN to include the ASCII FORTRAN structured IF statements. Modifying DMATRAN provided an automatic indentation capability for ASCII FORTRAN as well as DMATRAN programs.

ASCII FORTRAN (FORTRAN 77) adds 12 new statements to FORTRAN V, modifies six existing statements in FORTRAN V, and adds a character string data type and three operators related to that data type. GRC modified FAVS to recognize and correctly process the following new statement types:

(a) Structured IF statements

```
IF (e) THEN
ELSE IF (e) THEN
ELSE
END IF
```

(b) Character statements

```
CHARACTER [*len[,]name[,name]...]
V = e (character assignment)
```

(c) I/O statements

```
CLOSE (cllist)
INQUIRE (iflist)
INQUIRE (iulist)
OPEN (olist)
```

(d) Declaration statements

```
INTRINSIC fun [,fun]...
SAVE [a, [, a]...]
```

as well as the modified statement types:

(a) DO statement

DO S [,] i = e_1, e_2[, e_3]

(b) I/O statements

```
BACKSPACE (alist)
ENDFILE (alist)
READ (cilist) [iolist]
REWIND (ailist)
WRITE (cilist) [iolist]
```

Also all syntactically correct statements containing the new character operators .EQV., .NEQV., and // will be processed by FAVS.

The DMATRAN syntax for structured IF statements was modified to include the ASCII ELSE IF...THEN statement (the IF...THEN, ELSE, and ENDIF statements in DMATRAN are syntactically identical to those in ASCII FORTRAN). The DMATRAN precompiler was modified to automatically indent the new ELSE IF...THEN statement as well as to pass ASCII FORTRAN structured IF statements directly to the compiler without translating them into logically equivalent ANSI FORTRAN as is currently done.

5 USER INTERFACE ENHANCEMENTS

User interface enhancements to FAVS include simplifying FAVS commands, improving FAVS reports, and tailoring FAVS to the DMA UNIVAC 1100/80. Although no attempt has been made to quantify the benefits of these enhancements, it is evident that a more easily usable tool with readily understandable output will require less programmer time to perform the same analysis.

5.1 COMMANDS

Use of FAVS has been simplified by allowing most FAVS processing to be specified with UNIVAC @XQT parameters ("options" in UNIVAC terminology), and providing macro commands to generate individual reports. The FAVS options can still be specified on the command file by using the OPTIONS command described in Sec. 3.3 of the FAVS User's Manual. At DMA installations, the options can also be specified in the job control language by means of parameters following the UNIVAC @XQT command. One or more parameters can be specified at one run. Note that no commas are allowed between parameters. For example, if you want the STATIC analysis (S), the DOCUMENT reports (D), and INSTRUMENTation of the modules (I), the execute statement would be

```
@XQT,SDI R.FAVS
```

The FAVS option parameters are:

B	SUMMARY
D	DOCUMENT
E	EXPAND
I	INSTRUMENT
L	LIST
R	RESTART
S	STATIC
T	INPUT/OUTPUT
X	<described below>
Z	RESTRUCTURE

When these parameters are used, FAVS will not read the command file unless X is one of the parameters. The X tells FAVS that there will be other commands on the command file (refer to FAVS User's Manual for a discussion of commands).

New commands REPORT and FIRSTLINE were added to FAVS, and the LANGUAGE command was deleted.

The REPORT command selects specific reports to be produced during a FAVS run, when it is not desired to produce all the reports normally produced by the FAVS options. It has the same format as the OPTION command, i.e.:

REPORT = report list

Specified report names must be separated by commas. Blanks within the list are ignored. This command may appear within the command stream in any location that is valid for the OPTION command. The REPORT command cannot exceed 80 characters; continuation onto another card is not recognized. Instead a separate REPORT command should be given, or its parameters can be abbreviated.

The report names within the REPORT command are as follows:

REPORT NAME	MINIMUM ABBREVIATION	REPORT GENERATED
COMMONS	CO	Commons summary
PROFILE	PR	Statement profile
INVOCATIONS	L	Entries and invocations summary
COMMONS/ENHANCED	CO/E	Common matrices
BANDS/n	B or B/n	Invocation bands (n= number of levels)
SPACE	SP	Invocation space
SYMBOLS	SY	Symbol report
READS	R	I/O statements
CROSS	CR	Symbol cross reference
PICTURE	PI	Picture of module structure

The COMMONS and PROFILE reports are produced by the SUMMARY option; the INVOCATION report is included in both the SUMMARY and the DOCUMENT options. All the remaining reports, except PICTURE, are produced as part of the DOCUMENT option. The PICTURE report can only be obtained by using the REPORT command. It is not included in any of the options because the PICTURE report has limited use for DMATRAN source programs. Its primary function is to delineate the control flow of FORTRAN programs.

If the entire set of reports produced by an option is not desired, do not use the OPTION command; instead specify the appropriate report names in the REPORT command. If the same report is requested in both the OPTION and the REPORT command, the report will not be duplicated.

The FIRSTLINE command was added to FAVS in order to make FAVS easier to use on a UNIVAC 1100/80. When either the INSTRUMENT or RESTRUCTURE option is selected, the instrumented or restructured source program is written on UNIT 9. The user may use the command

FIRSTLINE = (<run stream command>)

to specify a UNIVAC run stream command that will be added as the first line of every element of the source program.

For example, when a FORTRAN source program is to be instrumented, then compiled and executed, the user could use the command,

FIRSTLINE = (@FOR,I TPF$.+).

FAVS will insert the UNIVAC command, @FOR,I TPF$.<element name> as the first line of each element (with the appropriate element name following TPF$.) If the UNIVAC FTN compiler is being used, the command could be

FIRSTLINE = (@FTN,I TPF$.+).

If the source code is written in DMATRAN a "C" should be substituted for the "@", because the DMATRAN precompiler must be used to

translate the DMATRAN code into FORTRAN before compilation by the FORTRAN compiler.[1] During the translation, the DMATRAN precompiler will automatically change the "C" to "@" and the element will be compiled by the compiler that is specified. The appropriate commands would be,

```
FIRSTLINE = (CFOR,I  TPF$.+).
```

for the UNIVAC FOR compiler and,

```
FIRSTLINE = (CFTN,I  TPF$.+).
```

for the UNIVAC FTN compiler.

Previously FAVS users were required to indicate explicitly (by the LANGUAGE command) when the DMATRAN structured dialect was being processed. Now FAVS accepts, as standard input, source written in UNIVAC FORTRAN V, UNIVAC ASCII FORTRAN, or the DMATRAN structured dialect. Explicit indication of the language dialect is no longer required.

5.2 MODIFIED REPORTS

Major changes were made to most FAVS reports during the FAVS Enhancement contract. All FAVS reports containing source text lines now display the original source text rather than harder to use reconstructed source text. The previous common matrix reports and library dependence matrix report proved to be hard to read (especially for large systems). They have been replaced by tabular reports and a common matrix for each common block rather than one matrix for all common blocks. The read statements report has been expanded into an I/O statements report. The monolithic cross-reference report was divided into common variable, external, and module cross-references. Splitting the static report into separate static and symbol reports improves the usefulness of FAVS static analysis. Restructuring now utilizes the graph checking option of STATIC to indicate unreachable statements (which will not be included in the restructured program).

---

[1] See DMATRAN User's Guide, General Research Corporation CR-1-673/1.

Statement Listing

Most source text editors are line-number-oriented or tag each source line with a unique name. It is essential that FAVS source reports contain the line numbers or names needed to edit the source text. FAVS users previously had to compare FAVS statement listing reports (Fig. 5.1) to source listings which did contain appropriate editing information. This can be an awkward and time-consuming task. Now, FAVS source statement listings (Fig. 5.2) contain appropriate editing information.

```
STATEMENT LISTING                SUBROUTINE EXAMPL ( INO. LENGTH )

 NO. LEVEL    LABEL    STATEMENT TEXT.                                    DDPATHS
-------------------------------------------------------------------------------
   1                    SUBROUTINE EXAMPL ( INFO. LENGTH )                ( 1)
   2          C
   3          C         ILLUSTRATION OF DMATRAN SYNTAX
   4          C
   5                    IF ( INFO .LE. 10 .AND. LENGTH .GT. 0 ) THEN      ( 2- 3)
   6 ( 1)               . CALL CALLER ( INFO )
   7                    ELSE
   8 ( 1)               . LENGTH = 50
   9                    ENDIF
  10                    CASEOF ( INFO + 6 )                               ( 4- 6)
  11                    CASE ( 14 )
  12 ( 1)               . LENGTH = LENGTH - INFO
  13                    CASE ( 17 )
  14 ( 1)               . DOWHILE ( INFO .LT. 20 )                        ( 7- 8)
  15 ( 2)               . . DOUNTIL ( LENGTH .LE. INFO )
  16 ( 3)               . . . INVOKE ( COMPUTE LENGTH )
  17 ( 3)               . . . IF ( LENGTH .GE. 30 ) THEN                  ( 9- 10)
  18 ( 4)               . . . . INVOKE ( PRINT-RESULTS )
  19 ( 3)               . . . ENDIF
  20 ( 2)               . . ENDUNTIL                                      ( 11- 12)
  21 ( 2)               . . INFO = INFO + 1
  22 ( 1)               . ENDWHILE
  23                    CASEELSE
  24 ( 1)               . DOWHILE ( LENGTH .GT. 0 )                       ( 13- 14)
  25 ( 2)               . . INVOKE ( COMPUTE LENGTH )
  26 ( 1)               . ENDWHILE
  27                    ENDCASE
  28                    BLOCK ( PRINT-RESULTS )                           ( 15)
  29 ( 1)               . WRITE ( 6, 1 ) INFO, LENGTH
  30 ( 1)      1        . FORMAT (10X,I5,20X,I5)
  31                    ENDBLOCK
  32                    BLOCK ( COMPUTE LENGTH )
  33 ( 1)               . LENGTH = LENGTH - 10
  34                    ENDBLOCK
  35                    RETURN
  36                    END
-------------------------------------------------------------------------------
```

This report, output for each module submitted to FAVS, contains the enhanced module listing with statement numbers, nesting levels, and DD-path numbers (at procedure entry and at each conditional statement).

Figure 5.1. FAVS.0 Statement Listing

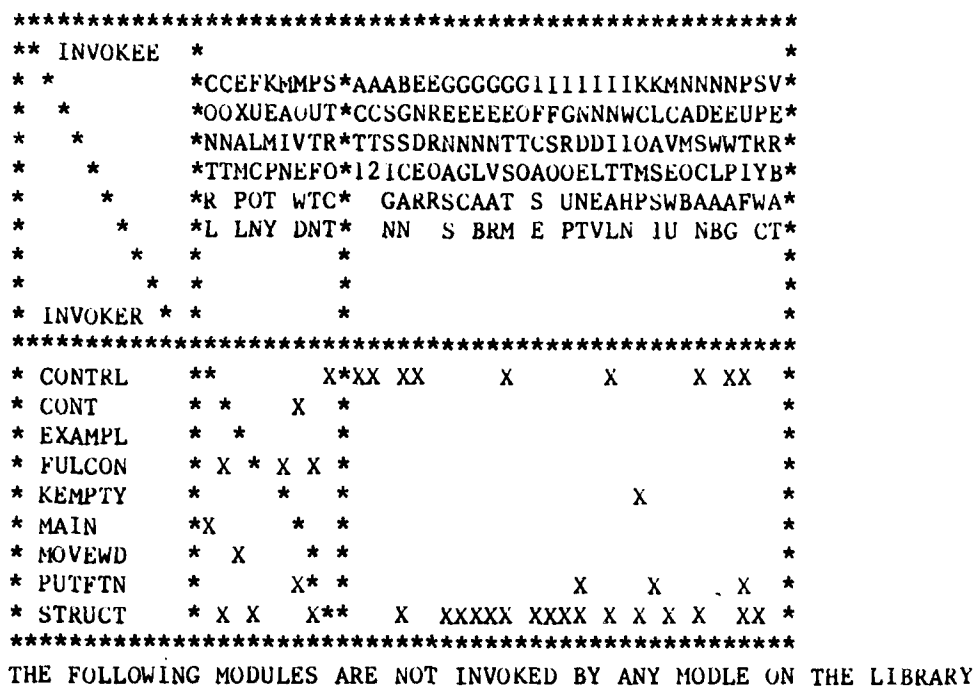
```
STATEMENT LISTING                    SUBROUTINE EXAMPL (INFO,LENGTH)

 STMT NEST  LINE  SOURCE...                                          ...SOURCE TAB
-------------------------------------------------------------------------------
    1          1       SUBROUTINE EXAMPL (INFO,LENGTH)
               2  C                                                  EXAMPL2
               3  C    ILLUSTRATION OF DMATRAN SYNTAX                EXAMPL3
               4  C                                                  EXAMPL4
    2          5       IF (INFO.LE.10 .AND. LENGTH.GT.0)THEN         EXAMPL5
    3    1     6       .   CALL CALLER ( INFO )                      EXAMPL6
    4          7       ELSE                                          EXAMPL7
    5    1     8       .   LENGTH=50                                 EXAMPL8
    6          9       END IF                                        EXAMPL9
    7         10       CASE OF (INFO+6)                              EXAMPL10
    8         11       CASE (14)                                     EXAMPL11
    9    1    12       .   LENGTH=LENGTH-INFO                        EXAMPL12
   10         13       CASE (17)                                     EXAMPL13
   11    1    14       .   DO WHILE (INFO.LT.20)                    EXAMPL14
   12    2    15       .   .   DO UNTIL (LENGTH.LE.INFO)            EXAMPL15
   13    3    16       .   .   .   INVOKE (COMPUTE LENGTH)          EXAMPL16
   14    3    17       .   .   .   IF (LENGTH.GE.30) THEN           EXAMPL17
   15    4    18       .   .   .   .   INVOKE (PRINT-RESULTS)       EXAMPL18
   16    3    19       .   .   .   END IF                           EXAMPL19
   17    2    20       .   .   END UNTIL                            EXAMPL20
   18    2    21       .   .   INFO=INFO+1                          EXAMPL21
   19    1    22       .   END WHILE                                EXAMPL22
   20         23       CASE ELSE                                     EXAMPL23
   21    1    24       .   DO WHILE (LENGTH.GT.0)                   EXAMPL24
   22    2    25       .   .   INVOKE (COMPUTE LENGTH)              EXAMPL25
   23    1    26       .   END WHILE                                EXAMPL26
   24         27       END CASE                                      EXAMPL27
   25         28       BLOCK (PRINT-RESULTS)                         EXAMPL28
   26    1    29       .   WRITE (6,1)INFO,LENGTH                   EXAMPL29
   27    1    30     1 .   FORMAT (10X,I5,20X,I5)                   EXAMPL30
   28         31       END BLOCK                                     EXAMPL31
   29         32       BLOCK (COMPUTE LENGTH)                        EXAMPL32
   30    1    33       .   LENGTH = LENGTH -10                      EXAMPL33
   31         34       END BLOCK                                     EXAMPL34
   32         35       RETURN                                        EXAMPL35
   33         36       END                                           EXAMPL36
-------------------------------------------------------------------------------
```

This report contains the indented module listing with statement numbers, source line numbers, and nesting levels.

Figure 5.2. FAVS.4 Statement Listing

Invocation Summary

The library dependence matrix (Fig. 5.3) was replaced with the invocation summary report (Fig. 5.4), which corrects several deficencies in the library dependence matrix and can be easily understood even when a large number of modules are involved. The library dependence matrix report was limited to 100 modules and it did not appear advisable to increase this limit and retain the matrix format, since the number of pages required to print the matrix would go up as the square of the number of modules. The resultant complexity of the library dependence matrix reflects the matrix format rather than the software system being analyzed. With the new invocation summary report, the complexity increases in direct proportion to the complexity of the software being analyzed. The invocation summary also includes entry points within FORTRAN modules, which were omitted in the earlier report.



THE FOLLOWING MODULES ARE NOT INVOKED BY ANY MODLE ON THE LIBRARY

MAIN

THE FOLLOWING MODULES DO NOT INVOKE ANY MODULE ON THE LIBRARY

EXAMPL KEMPTY

The interaction of all modules on the data base library is shown in the first matrix. If the library contains all modules in the user's program, this report provides a concise, complete picture of the total internal module dependencies. If the library contains a subset of the total program, this report aids in determining what modules do not interact with the component and might be better suited for another component. The modules are listed in alphabetical order.

The modules in the second matrix are not resident on the library. If the library allegedly contains all modules in the program, the external modules should consist only of system routines. If the library contains a component of the total program, this report shows the module invocation interfaces to other externals.

Considering the modules on the library as a pyramid representing the invocation hierarchy of the modules, this report also identifies the "top" and "bottom" modules in the system.

Figure 5.3. FAVS.0 Library Dependence Matrix

```
INVOCATION SUMMARY

ENTRY          LISTS OF CALLS
-------------------------------------------------------------------------------

PUTLST    WHICH IS DEFINED IN  GETBLK
          IS CALLED BY - -NONE-
          AND CALLS    - GETFRG  MAKFRG  XMIT

PUTWRD    WHICH IS DEFINED IN  GETBLK
          IS CALLED BY - PUTBEF  PUTBOT
          AND CALLS    - GETFRG  MAKFRG  XMIT

XMIT      WHICH IS UNDEFINED
          IS CALLED BY - GETBLK  NEXT    PREV    PUTAT   PUTBEF  PUTBOT

THE FOLLOWING ENTRIES ARE NOT CALLED
                       GETBLK  GETLST  GETWRD  ISRTAB  NEXT    PREV    PUTAT
```

This report shows the dependencies of the modules in the library by listing all modules which call an entry point and all calls from that entry point. If an entry is defined as an entry point within a module, the name of that module is indicated. This report includes all modules and entries on the restart file. An updated version of the report may be obtained by reanalyzing all changed modules and using the EXPAND option. The actual statements where invocations to a given entry point occur can be found in the externals cross reference report.

Figure 5.4. FAVS.4 Invocation Summary

Common Summary

The common matrix report (Fig. 5.5) was replaced by the common summary report (Fig. 5.6). The rationale for this change was similar to that discussed above. The number of pages necessary to produce this report reflects the matrix format rather than the complexity of the system analyzed. The tabular format of this report allows the 100 module limit to be eliminated.

C	**		*					.						*
O	*	* MODULE	*	C	C	E	F	K.	M	M	P	S		*
M	*	*	*	O	O	X	U	E.	A	O	U	T		*
M	*	*	*	N	N	A	L	M.	I	V	T	R		*
O	*	*	*	T	T	M	C	P.	N	E	F	U		*
N	*	*	*	R		P	O	T.		W	T	C		*
	*	*	*	L		L	N	Y.		D	N	T		*
N	*	*	*					.						*
O	*	COMMON *	*					.						*
	*		**					.						*
1	*	ACCTNG	*	X				.						*
2	*	CARDS	*	X				X.						*
3	*	CONSTN	*	X	X			X.			X	X		*
4	*	FORTRN	*	X	X			.			X	X		*
5	*	INTERN	*	X			X	.			X	X		*
6	*	INVOKE	*	X				.				X		*
7	*	RECN1Z	*	X				.						*
8	*	SESE	*	X				.						*
9	*	STACK	*	X				.				X		*
10	*	STATE	*	X			X	.			X	X		*
11	*	STYPE	*	X				.				X		*
12	*	TRACE	*	X				.			X			*
13	*	USEOPT	*	X	X			X.			X	X		*
14	*	WARNIN	*	X				.						*

This report listed all modules and all common blocks encountered. An "X" indicates the presence of that common in a module.

Figure 5.5. FAVS.0 Commons Matrix

```
COMMON SUMMARY

COMMON      MODULES WHICH INCLUDE THE COMMON
-------------------------------------------------------------------------

AISTO       MAKTAB

ALPHA       CMATRX

ANSI        CMATRX REFVAR

BLKSTO      DEPVOK

DBGCOM      MAKTAB

DEPCOM      DEPBND DEPVOK

EPT         DEPGRP DEPVOK REFVAR

FILES       CMATRX DEPBND DEPGRP DEPVOK XREFER

GLOBAL      DEPBND

HALPHA      XREFER

HCHARS      DEPGRP XREFER

HDIGIT      CMATRX

ICMMDS      STEP15

KOELMS      DEPVOK

MACHNE      DEPVOK

MDB         DEPVOK REFVAR

MMRY15      STEP15

MTHSTO      DEPGRP DEPVOK

MTHST1      DEPVOK XREFER
```

This report lists all modules and all common blocks encountered.

Figure 5.6. FAVS.4 Common Summary

Common Matrix (Enhanced)

The earlier FAVS common matrix (enhanced) (Fig. 5.7) suffered from the same artificial complexity problems as the other matrix reports. In addition, the use of a common number to identify the common block a variable was defined in led to considerable confusion. A report with more detail than the common summary, but less detail than the common cross reference, was desired. The solution to these concerns was a separate matrix for each common block. This matrix displays the use of common variables within a set of related modules (which all use some variable in the common). An example of the new common matrix report is shown in Fig. 5.8.

LIBRARY COMMON BLOCK MATRIX

```
-------------------------.---------
 C **           *         .         *
 O * * MODULE   * C C E F K.M M P S *
 M *  *         * O O X U E.A C D T *
 M *   *        * N N A L M.I V T R *
 O *    *       * T T M C P.N E F U *
 N *     *      * R   P O T.  N T C *
   *      *     * L   L N Y.  C N T *
 N *       *    *         .         *
 O * COMMON *   *         .         *
   *           **         .         *
-------------------------.---------
 1  * ACCTNG    * O         .         *
 2  * CARDS     * X        0.         *
 3  * CONSTN    * O X      X.     X X *
 4  * FORTRN    * O X       .     X O *
 5  * INTERN    * X     O   .     X O *
 6  * INVOKE    * O         .       X *
 7  * RECNIZ    * O         .         *
 8  * SESE      * O         .         *
 9  * STACK     * O         .       X *
 10 * STATE     * X     X   .     O X *
 11 * STYPE     * O         .       O *
 12 * TRACE     * X         .     X   *
 13 * USEOPT    * X O      O.     X X *
 14 * WARNIN    * O         .         *
------------------------- ---------
```

LEGEND

```
---------------------------------------------

COMMONS VS. MODULES
-------------------
X = > AT LEAST ONE SYMBOL REFERENCED
O = > NO SYMBOL EVER REFERENCED

SYMBOLS VS. MODULES
-------------------
X = > SYMBOL SET AND USED
O = > SYMBOL NEVER SET OR USED
S = > SYMBOL SET ONLY
U = > SYMBOL USED ONLY
E = > SYMBOL EQUIVALENCED (OVERLAID) ONLY
A = > SYMBOL IS AN ARRAY
```

LIBRARY COMMON SYMBOL MATRIX

```
-------------------------.---------
 C **           *         .         *
 O * * MODULE   * C C E F K.M M P S *
 M *  *         * O O X O E.A C U T *
 M *   *        * N N A L M.I V T R *
 O *    *       * T T M C P.N E F U *
 N *     *      * R   P O T.  W T C *
   *      *     * L   L N Y.  C N T *
 N *       *    *         .         *
 O * SYMBOL *   *         .         *
   *           **         .         *
-------------------------.---------
 2  * IECF      * O        O.         *
 13 * INDON     * O O      O.     O O *
 9  * INSTAK    * O         .       X *
 10 * ITYPE     * O     O   .     O U *
 A4 * KABEL     * C X       .     U O *
 4  * KENGTH    * O S       .     X O *
 A4 * KFTN      * O U       .     U O *
 13 * KOMFTN    * O O      O.     O U *
 5  * KSTMT     * O     O   .     O O *
A10 * LABEL     * O     X   .     O U *
 5  * LBK       * O U      U.     L U *
 10 * LENGTH    * S     O   .     O O *
 10 * LINBEG    * U     O   .     O O *
 10 * LINEND    * O     O   .     O O *
A10 * LIST      * O     O   .     O U *
 10 * LPOINT    * O     U   .     O U *
 A9 * LSTACK    * O         .       X *
 10 * LTYPE     * X     O   .     O U *
 13 * LUNFOR    * O O      O.     O O *
 13 * LUNOUT    * O O      O.     O O *
 10 * MENGTH    * O     O   .     O U *
 12 * NALTER    * O         .     O   *
 A6 * NAME1     * O         .       U *
 5  * NFATER    * O     O   .     O O *
 13 * NINDNT    * O O      O.     X O *
 5  * NLINES    * S     O   .     O O *
 6  * NOBE      * O         .       X *
 A6 * NOBLOK    * O         .       S *
 A6 * NOINV     * O         .       X *
 10 * NSTATE    * X     O   .     O O *
------------------------- ---------
```

Two matrices are produced by this report. The first one lists all common blocks encountered in any one of the modules in the set which was analyzed. If at least one symbol was used, it is indicated with an "X". If no symbol was ever referenced in the module, this is indicated by a "0". Routines from which a common block may safely be removed are easily found.

The second matrix lists only the symbols which are used by some module; the number of the common block in which it is found is printed to the left and corresponds to the number given to the common block in the first matrix. This report is an excellent aid when changes are being made to a software system.

Figure 5.7. FAVS.0 Common Matrix (Enhanced)

COMMON MATRICES

LEGEND (C=FIRST USED IN A CALL,E=EQUIVALENCED,S=SET,U=USED,X=SET AND USED)

```
        **           *             *
        * * MODULE   * N F F F F.G I *
        *   *        * E R U U U.E S *
        *     *      * X E T T T.T R *
        *       *    * T V A B B.B T *
        *         *  *     T E O.L A *
        *           **       F T.K B *
        *            *            .  *
COMMON  * SYMBOL  *  *            .  *
        *           **            .  *

AISTO   * FLCXXX     * U U U U U.U U *
        * FNUXXX     * U U U U U.U U *
        * FRGXXX     * U U U U U.U U *
        * FSZXXX     *            .U C *
        * ICHXXX     *     S S S.S   *
        * IXXXXX     * X X X X X.X X *
        * LNGXXX     * U U U U U.U U *
        * MAXXXX     *            .X   *
```

```
        **           *         *
        * * MODULE   * B F F S *
        *   *        * I A T D *
        *     *      * D S R B *
        *       *    * S T S A *
        *         *  * D W T S *
        *           ** R O R A *
        *            *         *
COMMON  * SYMBOL  *  *         *
        *           **         *

SDB     * ISCODE     * X X U S *
        * ISCONT     * U X     *
        * ISINDT     *     S   *
        * ISINFO     * S S   X *
        * ISLABL     * E E E E *
        * ISLONG     * X X   X *
        * ISPTR      *   X S X *
        * ISTYPE     * S X U X *
```

The common matrices report lists symbols which are used by at least one of the modules on the restart file. The symbol usage is explained in the legend at the top of the report; a blank space indicates that the symbol is not used in any way in that particular module. The symbols within each common are listed alphabetically in this report. Only modules which use at least one variable of a common block will occur in the matrix for that common. This report includes all commons and modules on the restart file. An updated version of this report may be produced by reanalyzing all changed modules and using the EXPAND option. When all modules in a software system have been entered onto a RESTART file, this report can be used to check for global set/use inconsistencies. A row of one or more U's indicates that a common variable is used but not set. A row of one or more S's indicates a common variable which is set but not used. A common variable which is not included in the matrix is never referenced in an executable statement. The statement number where common variables are referenced can be found in the common variable cross reference report.

Figure 5.8. FAVS.4 Common Matrices

Static Report

The use of static analysis as a consistency checking tool was emphasized by splitting the former static analysis report (Fig. 5.9) into a static analysis report (Fig. 5.10) which contains all consistency diagnostics after an appropriate source line, and a separate symbols report (Fig. 5.11) for symbol table information. The new static analysis report contains actual source text lines as read in, summarizes unknown externals at the end, and provides a complete cross-reference for variables which may have set/use inconsistencies. The symbol report is now alphabetically ordered. It omits the first statement, last statement, and total uses information, which is better described in the module cross-reference report (Fig. 5.14). In addition, an abbreviated form of the static analysis report is available (Fig. 5.12) which lists only the source lines related to inconsistencies.

```
STATIC ANALYSIS              SUBROUTINE CIRCLE ( AREA )

SEQ NEST SOURCE                                                                    UNKNOWN EXTERNALS
-------------------------------------------------------------------------------------------------
  1              SUBROUTINE CIRCLE ( AREA )
  2              COMMON / VALUES / DIAMTR
  3              INTEGER AREA
  4              RADIUS = DIAMTR / 2
  5              AREA = PI * RADIUS ** 2
                 ---------------------------------------------------------
                 -                        MODE WARNING                    -
                 - LEFT HAND SIDE HAS MODE INTEGER RIGHT HAND SIDE HAS MODE REAL -
                 ---------------------------------------------------------
  6              IF ( AREA .GT. 50 ) THEN
  7 ( 1)         .  CALL PRNT ( AREA )
                 ---------------------------------------------------------
                 -                        CALL ERROR                      -
                 -      PRNT      CALLED WITH 1 ACTUALLY HAS 2 ARGUMENTS    -
                 ---------------------------------------------------------
                 ---------------------------------------------------------
                 -                        CALL ERROR                      -
                 -PARAMETER  1 OF PRNT   .ACTUAL PARAMETER HAS MODE INTEGER -
                 -                       .FORMAL PARAMETER HAS MODE REAL   -
                 ---------------------------------------------------------
  8              END IF
  9              RETURN
 10              CALL STACK ( RADIUS, AREA )
                 ---------------------------------------------------------
                 -                        GRAPH WARNING                   -
                 -       STATEMENT 10 IS UNREACHABLE OR IS IN AN INFINITE LOOP -
                 ---------------------------------------------------------
                                                                                   STACK
 11              END
-------------------------------------------------------------------------------------------------
                 STATEMENT ANALYSIS SUMMARY      ERRORS     WARNINGS
                 --------------------------      ------     --------
                 GRAPH CHECKING                       0            1
                 CALL CHECKING                        2            0
                 MODE CHECKING                        0            1

                                    1ST    TOTAL    LAST    IN/OUT    ACTUAL     PHYSICAL
NAME        SCOPE        MODE      STMT   USES     STMT    USE       USE        UNITS
-------------------------------------------------------------------------------------------------
AREA        PARAMETER    INTEGER   1      6        10                BOTH

DIAMTR      VALUES       REAL      2      2        4                 INPUT

RADIUS      LOCAL        REAL      4      3        10

PI          LOCAL        REAL      5      1        5
                 ---------------------------------------------------------
                 -                     SET/USE WARNING                    -
                 - VARIABLE PI    MAY BE USED BEFORE BEING ASSIGNED A VALUE -
                 ---------------------------------------------------------
-------------------------------------------------------------------------------------------------
                 SYMBOL ANALYSIS SUMMARY         ERRORS   WARNINGS
                 -----------------------         ------   --------
                 SET/USE CHECKING                     0          1
```

The Statement Analysis Summary contained the warning and error messages interspersed appropriately in the code. Unknown externals, routines called which were not in the set submitted to FAVS, were listed on the right side of the printout. A tabulation of the errors and warnings was listed at the bottom.

The Symbol Analysis Summary showed the name, scope, and mode of each symbol in any executable statement in the module. The actual use of global variables was defined as INPUT, OUTPUT, or BOTH. For any variable that was used before being assigned a value or set and not used, a warning indicated the condition which could lead to errors.

Figure 5.9. FAVS.0 Static Analysis Report

```
STATIC ANALYSIS                    SUBROUTINE CIRCLE ( AREA )

 STMT NEST  LINE  SOURCE...                                          ...SOURCE TAB
-------------------------------------------------------------------------------
    1          1      SUBROUTINE CIRCLE ( AREA )
    2          2      INTEGER AREA
    3          3      DATA PI / 3.1416 /
    4          5      INPUT (/R/ RADIUS )
    5          6      RADIUS = DIAMTR / 2
                    ------------------------------------------------------------
                    -                        SET/USE ERROR                      -
                    - VARIABLE DIAMTR     USED BUT NEVER SET      REFER TO STATEMENT(S)-
                    -  5
                    ------------------------------------------------------------
    6          7      AREA = PI * RADIUS**2
                    ------------------------------------------------------------
                    -                        MODE WARNING                       -
                    - LEFT HAND SIDE HAS MODE INTEGER   RIGHT HAND SIDE HAS MODE REAL -
                    ------------------------------------------------------------
    7          8      IF ( AREA .GT. 50 ) THEN
    8    1     9      .   CALL PRINT ( AREA )
                    ------------------------------------------------------------
                    -                        MODE WARNING                       -
                    -PARAMETER  1 OF PRINT      ACTUAL PARAMETER HAS MODE INTEGER -
                    -                           FORMAL PARAMETER HAS MODE REAL    -
                    ------------------------------------------------------------
                    ------------------------------------------------------------
                    -                        CALL ERROR                         -
                    -         PRINT      CALLED WITH  1 ACTUALLY HAS  2 ARGUMENTS -
                    ------------------------------------------------------------
    9         10      END IF
   10         11      OUTPUT (/R/ AREA )
   11         13      RETURN
   12         14      CALL STACK ( RADIUS, AREA )
                    ------------------------------------------------------------
                    -                        GRAPH WARNING                      -
                    -       STATEMENT  12  IS UNREACHABLE OR IS IN AN INFINITE LOOP -
                    ------------------------------------------------------------
   13         15      END
-------------------------------------------------------------------------------

                    STATIC   ANALYSIS   SUMMARY      ERRORS  WARNINGS
                    ---------------------------      ------  --------
                    GRAPH CHECKING                        0         1
                    CALL CHECKING                         1         0
                    MODE CHECKING                         0         2
                    SET/USE CHECKING                      1         0
          CALL CHECKING WAS NOT PERFORMED FOR THE FOLLOWING UNKNOWN EXTERNALS ...
          STACK
```

The Static Analysis Summary contains the warning and error messages interspersed appropriately in the code. Unknown externals (routines called which are not in the set submitted to FAVS) are listed at the bottom of the report. A tabulation of the errors and warnings is listed at the bottom.

Figure 5.10. FAVS.4 Static Analysis Report (with LIST option)

```
SYMBOLS                                  SUBROUTINE SUBASA ( MODULE, ISTMT, IRETRN )

  NAME      SCOPE        TYPE       MODE       USE        OTHER INFORMATION...
-------------------------------------------------------------------------------
 KXPAR     KDELMS     VARIABLE   INTEGER    USED
 KXPARM    RPTCOM     VARIABLE   INTEGER    USED
 KXTPLS    RPTCOM     VARIABLE   INTEGER    USED
 LIST      MTHSTO     ARRAY      INTEGER    SET/USED
 MARGS     MDB        VARIABLE   INTEGER    SET
 MBLOKS    MDB        VARIABLE   INTEGER    EQUIV
 MBRCHN    MTHTYP     VARIABLE   INTEGER    USED
 MCALL     MTHTYP     VARIABLE   INTEGER    USED
 MCIO      MTHTYP     VARIABLE   INTEGER    USED
 MCMMNS    MDB        VARIABLE   INTEGER    SET/USED
 MDUM26    (LOCAL)    VARIABLE   INTEGER    EQUIV
 MENTR     MTHTYP     VARIABLE   INTEGER    USED
 MENTRS    MDB        VARIABLE   INTEGER    SET/USED
 MENTR2    MTHTYP     VARIABLE   INTEGER    USED
 MLGLS     (LOCAL)    VARIABLE   INTEGER    SET/USED
 MEQUVS    MDB        VARIABLE   INTEGER    SET/USED
 MEXEC     MTHTYP     VARIABLE   INTEGER    USED
 MEXIT     MTHTYP     VARIABLE   INTEGER    USED
 MGOTO     MTHTYP     VARIABLE   INTEGER    USED
 MIF       MTHTYP     VARIABLE   INTEGER    USED
 MJUNCT    MTHTYP     VARIABLE   INTEGER    USED
 MMODE     MDB        VARIABLE   INTEGER    SET/USED
 MNAME     MDB        VARIABLE   INTEGER    EQUIV
 MNONX     MTHTYP     VARIABLE   INTEGER    USED
 MODULE    PARAMETER  VARIABLE   INTEGER
 MPRSET    MTHTYP     VARIABLE   INTEGER    USED
 MREAD     MTHTYP     VARIABLE   INTEGER    USED
 MREADS    MDB        VARIABLE   INTEGER    SET/USED
 MTYPE     MDB        VARIABLE   INTEGER    SET
 MWRITS    MDB        VARIABLE   INTEGER    SET/USED
 NONEXS    (LOCAL)    ARRAY      INTEGER    SET/USED
 NUMEXS    (LOCAL)    VARIABLE   INTEGER    SET/USED
 NUMNON    (LOCAL)    VARIABLE   INTEGER    SET/USED

      THE FOLLOWING LOCAL VARIABLES WERE DEFINED BUT NOT USED...
      TOKADD

      THE FOLLOWING NONLOCAL VARIABLES ARE SET...
      IRETRN  MTYPE   MMODE   MCMMNS  MENTRS  MARGS   MEQUVS  MREADS  MWRITS  ISTYPE  ISCODE
      ISINFO  LIST
```

This report is generated for each module analyzed during a FAVS run. The symbols are ordered alphabetically, and symbols which are only defined and never referenced are not included. Symbols which have the scope (LOCAL) are known only within the module being reported on. Symbols with the scope parameter are formal parameters for the module. All other scope classifications indicate the name of the common block the common variables are defined in. Each symbol is either of type variable or array, and of mode integer, real, logical, character, complex, or double precision. The use column provides a summary of how the symbol is used in the module. Local symbols which were defined but not referenced and all non-local variables (parameters and common variables) which are set within the module are noted at the end of the report.

Figure 5.11. FAVS.4 Symbols Report

```
STATIC ANALYSIS                    SUBROUTINE CIRCLE ( AREA )

 STMT NEST  LINE  SOURCE...                                                ...SOURCE TAB
-------------------------------------------------------------------------------------
    5          6      RADIUS = DIAMTR / 2
                     ------------------------------------------------------------------
                     -                        SET/USE ERROR                           -
                     - VARIABLE DIAMTR     USED BUT NEVER SET        REFER TO STATEMENT(S)-
                     -  5
                     ------------------------------------------------------------------
    6          7      AREA = PI * RADIUS**2
                     ------------------------------------------------------------------
                     -                        MODE WARNING                            -
                     - LEFT HAND SIDE HAS MODE INTEGER   RIGHT HAND SIDE HAS MODE REAL -
                     ------------------------------------------------------------------
    8          9      CALL PRINT ( AREA )
                     ------------------------------------------------------------------
                     -                        MODE WARNING                            -
                     -PARAMETER  1 OF PRINT      ACTUAL PARAMETER HAS MODE INTEGER    -
                     -                           FORMAL PARAMETER HAS MODE REAL       -
                     ------------------------------------------------------------------
                     ------------------------------------------------------------------
                     -                        CALL ERROR                              -
                     -         PRINT      CALLED WITH  1 ACTUALLY HAS  2 ARGUMENTS    -
                     ------------------------------------------------------------------
   12         14      CALL STACK ( RADIUS, AREA )
                     ------------------------------------------------------------------
                     -                        GRAPH WARNING                           -
                     -        STATEMENT  12  IS UNREACHABLE OR IS IN AN INFINITE LOOP -
                     ------------------------------------------------------------------
-------------------------------------------------------------------------------------
                      STATIC   ANALYSIS   SUMMARY      ERRORS  WARNINGS
                      ------------------------------   ------  --------
                      GRAPH CHECKING                        0         1
                      CALL CHECKING                         1         0
                      MODE CHECKING                         0         2
                      SET/USE CHECKING                      1         0
     CALL CHECKING WAS NOT PERFORMED FOR THE FOLLOWING UNKNOWN EXTERNALS ...
     STACK
```

This report is an abbreviated version of the Static Analysis report generated when the STATIC and LIST options are both used (Fig. 5.10). Only the statements that cause errors or warnings are listed in the Static Analysis report when the STATIC option is specified without the LIST option.

Figure 5.12. FAVS.4 Static Analysis Report (without LIST option)

Cross Reference

The single FAVS.0 cross reference report (Fig. 5.13) has been divided into a cross reference for each module (Fig. 5.14), a common variable cross reference (Fig. 5.15), and an external cross reference (Fig. 5.16). The earlier cross reference did not indicate whether a symbol was an external, a common variable, or a variable local to a particular module. Now this information is clearly shown in the three new cross references.

```
CROSS REFERENCE

GENERAL CROSS REFERENCE LISTING

MODULES INCLUDED --
        CONTRL
        CONT
        EXAMPL
        FULCON
        KEMPTY
        MAIN
        MOVEWD
        PUTFTN
        STRUCT

SYMBOL    MODULE    USED/SET/DEFINITION ( * INDICATES SET, D INDICATES DEFINITION )
ACT1      CONTRL    172
ACT2      CONTRL    174
ASSIGN    STRUCT    180
BGSCAN    CONTRL    168
CONTRL    CONTRL      1
          MAIN        2
CONT      CONT        1
          FULCON     14
          STRUCT     86   103   124   153   165   202   236   258   262   292   303   306   345
ENDER     CONTRL    183
ERROR     STRUCT     53   107   111   113   128   130   169   171   213   217   219   240   244
EXAMPL    EXAMPL      1
          MOVEWD     33
FULCON    FULCON      1
          STRUCT     84   101   122   137   160   199   234   255   275   298
GENASS    STRUCT    341
GENGO     STRUCT    369
GENLAB    STRUCT     73    81    85    98   102   123   139   141   149   152   161   164   195
                    281   283   291   299   302   305   339   340   357   360   371
GENVAR    STRUCT    179   208
GETSTM    CONTRL    164
GOTO      STRUCT     82    99   150   162   196   232   278   300   343   358
IARRY1    MOVEWD      1   23D   29*
IARRY     MOVEWD      1   22D   29
ICONT     CONT       24D   25D   25D   25D   25D   25D   25D   25D   25D   28
IEOF      CONTRL     29D   165   180
          KEMPTY      5D
IERROR    STRUCT     92*   93    94*   95   110   120*  121   127   158*  159   168   190*  191
                    243   253*  254.  265   296*  297   309
```

This report provided a symbol cross reference listing for all modules on the library. The symbol types where variables, file names, block names, and subprogram names. Adjacent to the statement number of the symbols appearance was a flag "*", (or D) which indicated setting or definition.

Figure 5.13. FAVS.0 Cross Reference Report

CROSS REFERENCE SUBROUTINE SDBASA (MODULE, ISTMT, IRETRN)

NAME	SCOPE	MODULE	USED/SET/EQUIVALENCED (* INDICATES SET)
ADDEPT	EXTERNAL	SDBASA	65 84 250
CALLED	(LOCAL)	SDBASA	24* 274 275*
ERROR	EXTERNAL	SDBASA	151
I	(LOCAL)	SDBASA	48* 49 50 51* 51 75* 76 77 78* 78 120* 121
			145 146 161* 162 171* 172 172 173* 173 175 177 191*
			227 228 342* 343 343 344* 344 346 353* 354 354 355*
IAGT	ANSI	SDBASA	149
IBAPAR	EXTERNAL	SDBASA	115 170 210
ICGT	ANSI	SDBASA	147
ICOMAS	(LOCAL)	SDBASA	119* 123* 123 127
IDUM	(LOCAL)	SDBASA	37 54 159 169
IDX	(LOCAL)	SDBASA	32* 33 33 33 34* 34 36 41 41 43 49 56
			76 80* 80 82* 82 84
IEND	ANSI	SDBASA	307
IENT	ANSI	SDBASA	246 316
IEXECS	(LOCAL)	SDBASA	307* 308* 309* 310* 311* 312* 313* 314* 315* 316* 317* 318*
			327* 328* 329* 330* 331* 332* 333* 334* 335* 336* 337* 354
IFT	ANSI	SDBASA	113 308
IF2	ANSI	SDBASA	128
IF3	ANSI	SDBASA	131
IGT	ANSI	SDBASA	136 311
IGTTOK	EXTERNAL	SDBASA	137
IMDB	(LOCAL)	SDBASA	10E
IPAR	(LOCAL)	SDBASA	115* 116 117 120 170* 172 175 180* 182 182 184
IREADC	FTNEXT	SDBASA	235
IRET	ANSI	SDBASA	188 315
IRETRN	PARAMETE	SDBASA	25* 269*
ISCLAS	EXTERNAL	SDBASA	138
ISCODE	SDB	SDBASA	29* 67* 89* 97* 112* 114* 118* 140* 155* 159* 167* 176*
ISDB	(LOCAL)	SDBASA	16E
ISEXEC	(LOCAL)	SDBASA	111 358* 360*
ISINFO	SDB	SDBASA	31* 129* 132* 139* 169* 199* 204* 210* 212 215 228* 249*
ISLABL	SDB	SDBASA	16E
ISLONG	SDB	SDBASA	33 36 37 41 59 70 73 115 116 121 139 142
			192 206* 206 210 224 227
ISNONX	(LOCAL)	SDBASA	88 347* 349*
ISPTR	SDB	SDBASA	117* 145* 177* 184* 185 216* 234 240
ISTMT	PARAMETE	SDBASA	65 84 103 250
ISTOP	ANSI	SDBASA	188 190 318
ISTYPE	SDB	SDBASA	27 27 27 42* 59 64 74* 90 93 96 99 100*
			166 188 188 190 198 203 208 208 208 208 208 208
			252 255 255 255 343 354
ITEM	(LOCAL)	SDBASA	137* 138
IVMODE	EXTERNAL	SDBASA	54
IWRTEC	FTNEXT	SDBASA	241
IXABNL	FTNEXT	SDBASA	302
IXASS	ANSI	SDBASA	252 317

This report provides a symbol cross reference for each module analyzed during a FAVS run. All local symbols, external symbols, and common symbols referenced in the module are included. Symbol names in the first column are ordered alphabetically. The scope column indicates symbols known only within this module (LOCAL), external symbols, and symbols which are defined in common blocks included in the module (all others). Statements (FAVS statement numbers) which use a symbol are followed by a blank, statements which set a symbol are followed by a '*', and equivalence statements containing the symbol are followed by an 'E'.

Figure 5.14. FAVS.4 Cross Reference (Individual Module)

CROSS REFERENCE

NAME	SCOPE	MODULE	USED/SET/EQUIVALENCED (* INDICATES SET)
AIDBG	DBGCOM	ISRTAB	82
CORE		GETBLK	26E
		ISRTAB	25E
		NEXT	18E
		PREV	18E
		PUTAT	18E
		PUTBEF	18E
		PUTBOT	18E
FLCXXX	AISTO	GETBLK	146 190 231 266
	AISTO	ISRTAB	67
	AISTO	NEXT	39
	AISTO	PREV	39
	AISTO	PUTAT	33
	AISTO	PUTBEF	43
	AISTO	PUTBOT	43
FNUXXX	AISTO	GETBLK	142 187 227 261
	AISTO	ISRTAB	62
	AISTO	NEXT	36
	AISTO	PREV	36
	AISTO	PUTAT	30
	AISTO	PUTBEF	40
	AISTO	PUTBOT	40
FRGDIR	POOLCM	GETBLK	149* 193* 240* 268*
	POOLCM	ISRTAB	79*
	POOLCM	NEXT	43*
	POOLCM	PREV	43*
	POOLCM	PUTAT	38*
	POOLCM	PUTBEF	50*
	POOLCM	PUTBOT	50*
FRGXXX	AISTO	GETBLK	148 192 240 268
	AISTO	ISRTAB	79
	AISTO	NEXT	43
	AISTO	PREV	43
	AISTO	PUTAT	38
	AISTO	PUTBEF	50
	AISTO	PUTBOT	50
FSZXXX	AISTO	GETBLK	140 141 185 186 216 219 220 248 249
	AISTO	ISRTAB	66
ICHXXX	AISTO	GETBLK	56* 81* 118* 168* 235*
	AISTO	PUTAT	35*
	AISTO	PUTBEF	47*
	AISTO	PUTBOT	47*
IXXXXX	AISTO	GETBLK	147* 147 149 191* 191 193 239* 239 240 267* 267 268
	AISTO	ISRTAB	78* 78 79
	AISTO	NEXT	42* 42 43
	AISTO	PREV	42* 42 43
	AISTO	PUTAT	37* 37 38

This multi-module report shows where variables in common blocks are used, set or equivalenced. The report is alphabetically ordered by the name of the common variable. The common block which contains the variable is indicated in the scope column. Modules which reference the common variable are alphabetically ordered in the module column. Statements (FAVS statement numbers) within each module are shown next. A blank following the statement number indicates the variable is used there, a '*' indicates the variable is set, and an 'E' indicates the variable is equivalenced. This report is produced for all modules and all commons on the restart file. An updated version may be obtained by reanalyzing all changed modules and using the EXPAND option. A summary of the information in this report is provided in the common matrices report.

Figure 5.15. FAVS.4 Cross Reference (Common Variables)

CROSS REFERENCE

NAME	SCOPE	MODULE	USED/SET/EQUIVALENCED (* INDICATES SET)			
EROR	EXTERNAL	ISRTAB	87			
FRELNK	EXTERNAL	PUTBEF	31			
	EXTERNAL	PUTBOT	29	31		
GETFRG	EXTERNAL	GETBLK	262			
	EXTERNAL	ISRTAB	63			
	EXTERNAL	NEXT	37			
	EXTERNAL	PREV	37			
	EXTERNAL	PUTAT	31			
	EXTERNAL	PUTBEF	41			
	EXTERNAL	PUTBOT	41			
IGTWRD	EXTERNAL	NEXT	31	33		
	EXTERNAL	PREV	31	33		
	EXTERNAL	PUTBEF	32			
	EXTERNAL	PUTBOT	35			
ITSFRG	EXTERNAL	ISRTAB	60			
	EXTERNAL	NEXT	35			
	EXTERNAL	PREV	35			
	EXTERNAL	PUTAT	29			
	EXTERNAL	PUTBEF	39			
	EXTERNAL	PUTBOT	39			
LGTMLT	EXTERNAL	ISRTAB	52	53		
MAKFRG	EXTERNAL	GETBLK	224			
MINO	EXTERNAL	ISRTAB	66			
PUTWRD	EXTERNAL	PUTBEF	33	35	37	
	EXTERNAL	PUTBOT	32	36	38	
XMIT	EXTERNAL	GETBLK	45	55	234	237
	EXTERNAL	NEXT	40	46		
	EXTERNAL	PREV	40	46		
	EXTERNAL	PUTAT	34			
	EXTERNAL	PUTBEF	44			
	EXTERNAL	PUTBOT	44			

This multi-module report shows the FAVS statement number where each external is referenced. The report is alphabetically ordered by the name of the external. Modules which reference the external are alphabetically ordered in the module column. Statements (FAVS statement numbers) within each module are shown in the next column. This report is produced for all modules on the restart file. An updated version may be obtained by reanalyzing all changed modules and using the EXPAND option. A summary of information contained in this report is provided by the Invocation Summary Report. The text of each invocation can be found by referring the FAVS statement listing or Invocation Report for each module. Note that these reports are not generated from the restart file but rather from source analyzed during a FAVS run.

Figure 5.16. FAVS.4 Cross Reference (Externals)

Restructure

The statement listing produced during restructuring has been enhanced to indicate any structurally unreachable statements. These statements will not be included in the restructured source. An example of this report is shown in Fig. 5.17.

```
STATIC ANALYSIS                    SUBROUTINE BSORT ( N, ARRAY )

 STMT NEST  LINE  SOURCE...                                        ...SOURCE TAB
-------------------------------------------------------------------------------
    1          1       SUBROUTINE BSORT ( N, ARRAY )
    2          2       DIMENSION ARRAY ( 100 )
    3          3       DO 1 I = 2, N
    4    1     4       .   IF ( ARRAY(I-1) .LE. ARRAY(I) ) GO TO 1
    6    1     5       .   SMALL = ARRAY(I)
    7    1     6       .   ARRAY(I) = ARRAY(I-1)
    8    1     7       .   J = I - 2
    9    1     8     2 .   IF ( J .LT. 1 ) GO TO 4
   11    1     9       .   IF ( SMALL .LT. ARRAY(J) ) GO TO 3
   13    1    10     4 .   ARRAY(J+1) = SMALL
   14    1    11       .   GO TO 1
   15    1    12     3 .   ARRAY(J+1) = ARRAY(J)
   16    1    13       .   J = J - 1
   17    1    14       .   GOTO 2
   18         15   1  CONTINUE
   19         16      RETURN
   20         17      END
-------------------------------------------------------------------------------

                    STATIC   ANALYSIS   SUMMARY      ERRORS   WARNINGS
                    ----------------------------     ------   --------
                    GRAPH CHECKING                        0          0
```

This report is a source listing of the original FORTRAN module. It is enhanced by indentation and statement and nesting level numbers.

Figure 5.17. Restructure Report

New Reports

Two new reports were added to FAVS. The picture report (Fig. 5.18) can be used to pictorally display the branching structure of large FORTRAN programs. It essentially combines a source listing and label cross reference into a single, easier to use report.

The second new report is the interface changes report. An INTERFACE CHANGES report is generated for each FAVS run. It lists each module name and indicates changes in interface properties such as parameters added or deleted, common blocks added or deleted, and external references added or deleted. It also lists calls to undefined entries and to entries or commons which are no longer used. Fig. 5.19 is an example of an INTERFACE CHANGES report. New modules are indicated by the words "new module" in the fourth column.

```
PICTURE                                   SUBROUTINE SORT(A,II,JJ)

     UPWARD JUMPS           STATEMENT TEXT           ( B=BEGIN, E=END, S=SELF LOOP )      (SHORT)  DOWNWARD JUMPS  (LONG)
ABCDEFGHIJKLMNOPQRST                                                                       ABCDEFGHIJKLMNOPQRSTUVWXYZ123456789
---------------------------------------------------------------------------------------------------------------------------
                            SUBROUTINE SORT(A,II,JJ)                                       B
                            DIMENSION A(1),IU(16),IL(16)                                   .
                            INTEGER A,T,TT                                                 .
                            M = 1                                                          .
                            I = II                                                         .
                            J = JJ                                                         .
              E          5  IF(I .GE. J) GO TO 70                                          EBB
              .                                                                            BE.
              E.        10  K = I                                                          .BE
              ..            IJ = (J + I)/2                                                 ..
              ..            T = A(IJ)                                                      ..
              ..            IF( A(I) .LE. T) GOTO 20                                       .EBB
              .                                                                            .BE.
              ..            A(IJ) = A(I)                                                   .. .
              ..            A(I) = T                                                       .. .
              ..            T = A(IJ)                                                      .. .
              ..        20  I = J                                                          .EBE
              ..            IF(A(J) .GE.T) GO TO 40                                        .BEB
              ..                                                                           .EB.
              ..            A(IJ) = A(J)                                                   . ..
              ..            A(J) = T                                                       . ..
              ..            T = A(IJ)                                                      . ..
              ..            IF( A(I) .LE. T) GO TO 40                                      .B.EB
              ..                                                                           .E.B.
              ..            A(IJ) = A(I)                                                   . ...
              ..            A(I) = T                                                       . ...
              ..            T = A(IJ)                                                      . ...
              ..            GO TO 40                                                       .B..E
             E..        30  A(L) = A(K)                                                    ....B
             ...            A(K) = TT                                                      .....
            E...        40  L = L - 1                                                      .EEEEB
            ....            IF(A(L) .GT. T) GO TO 40                                       .BB  E
            B...                                                                           .E.
             ...            TT = A(L)                                                      ..
            E...        50  K = K + 1                                                      .BE
            ....            IF( A(K) .LT. T ) GO TO 50                                     .EBB
            B...                                                                           . E.
             ...            IF( K .LE. L) GO TO 30                                         .BBE
             B..                                                                           .E.
              ..            IF( L-I .LE. J-K) GO TO 60                                     .BEB
              ..                                                                           .EB.
              ..            IL(M) = I                                                      . ..
              ..            IU(M) = L                                                      . ..
              ..            I = K                                                          . ..
              ..            M = M + 1                                                      . ..
              ..            GO TO 80                                                       .B.E
              ..        60  IL(M) = K                                                      ..EB
              ..            IU(M) = J                                                      .. .
              ..            J = L                                                          .. .
              ..            M = M + 1                                                      .. .
              ..            GO TO 80                                                       ..BE
             E..        70  M = M -1                                                       E..B
             ...            IF( M .EQ. 0) RETURN                                           B..EB
             ...                                                                           E..B.
             ...            I = IL(M)                                                      ....
             ...            J = IU(M)                                                      ....
             ...        80  IF(J-I .GE.11) GO TO 10                                        BEE.EB
              .B.                                                                          E  . .
             . .            IF( I .EQ. II) GO TO 5                                         BB . E
              . B                                                                          E. .
              .             I = I -1                                                       . .
             .EE        90  I = I + 1                                                      BE .
             ...            IF( I .EQ. J) GO TO 70                                         EBB.
             B..                                                                            E..
              ..            T = A(I+1)                                                     ..
              ..            IF( A(I) .LE. T ) GO TO 90                                     BBE.
              .B                                                                           E. .
              .             K = I                                                          . .
              .E       100  A(K+1) = A(K)                                                  BE .
              ..            K = K - 1                                                      . .
              ..            IF(T .LT. A(K)) GO TO 100                                      EBB.
              .B                                                                            E..
              .             A(K+1) = T                                                      ..
              B             GO TO 90                                                        E.
                            END                                                              E
---------------------------------------------------------------------------------------------------------------------------
```

The PICTURE report can only be obtained by using the REPORT=PICTURE command; it is not included in any of the options because the PICTURE report has limited use for DMATRAN source programs. The primary function of this report is to delineate the control flow of FORTRAN programs. The downward flows are shown on the right of the report. The upward flows are shown on the left. The B stands for the start of a path and the E stands for the end of a path. This report is especially helpful in breaking down large FORTRAN programs into smaller parts that are more manageable for FAVS to restructure. Since the PICTURE report shows the beginning and ending of paths, it helps the user determine which are logically cohesive sections of code. These sections of code can be bounded by the DMATRAN BLOCK - END BLOCK constructs in order to simplify restructuring and make the programs easier to use.

Figure 5.18. FAVS.4 Picture of Module Structure

INTERFACE CHANGES

MODULE	SYMBOL	TYPE OF CHANGE		(U USED,S SET,X SET/USED,*-UNKNOWN)
GETBLK			NEW MODULE	
	PUTBLK	ENTRY	NEW ENTRY	
	GETWRD	ENTRY	NEW ENTRY	
	PUTWRD	ENTRY	NEW ENTRY	
	GETLST	ENTRY	NEW ENTRY	
	PUTLST	ENTRY	NEW ENTRY	
	ATTNFO	COMMON	********	NO VARIABLE IN COMMON IS USED (EQUIVALENCES NOT CHECKED)
	DBGCOM	COMMON	********	NO VARIABLE IN COMMON IS USED (EQUIVALENCES NOT CHECKED)
	FILES	COMMON	********	NO VARIABLE IN COMMON IS USED (EQUIVALENCES NOT CHECKED)
TSKTAB			NEW MODULE	
	TABSIZ	COMMON	********	NO VARIABLE IN COMMON IS USED (EQUIVALENCES NOT CHECKED)
NEXT			NEW MODULE	
	TABDEF	COMMON	********	NO VARIABLE IN COMMON IS USED (EQUIVALENCES NOT CHECKED)
PREV			NEW MODULE	
	TABDEF	COMMON	********	NO VARIABLE IN COMMON IS USED (EQUIVALENCES NOT CHECKED)
PUTAT			NEW MODULE	
	TABDEF	COMMON	********	NO VARIABLE IN COMMON IS USED (EQUIVALENCES NOT CHECKED)
PUTBOT			NEW MODULE	
	TABDEF	COMMON	********	NO VARIABLE IN COMMON IS USED (EQUIVALENCES NOT CHECKED)
PUTBOT			NEW MODULE	
	TABDEF	COMMON	********	NO VARIABLE IN COMMON IS USED (EQUIVALENCES NOT CHECKED)

CALLS TO THE FOLLOWING UNDEFINED ENTRYS WERE ADDED...

ERROR FIRTNK GETERG TGTWRD TTSERG TGTMLT MAKERG MTRO XMIT

Figure 5.19. FAVS.4 Interface Changes Report

6 FAVS ERROR CORRECTION

Two types of errors were corrected during the FAVS software updates: Errors introduced in earlier updates, and residual errors present in FAVS after its initial installation at the DMA sites. This section discusses the residual errors.

6.1 FORTRAN V PROCS

UNIVAC FORTRAN V allows identical sections of source text to be inserted with the FORTRAN V INCLUDE statement. This statement refers to the name of a text section (PROC) of one or more lines, named and saved using the UNIVAC PDP processor. A PROC is preceded by a line which contains the name of the PROC beginning in the FORTRAN label field, one or more blanks or asterisks, and then the keyword "PROC" beginning after column 6. The original version of FAVS incorrectly looked for the keyword "PROC" followed by the name of the PROC. FAVS was corrected to look for the correct syntax.

6.2 FORTRAN V COMPILER STATEMENT

UNIVAC FORTRAN V allows compiler directives to be specified with a FORTRAN V compiler statement, which must precede the compilation unit to which it pertains. The original FAVS discarded all comments, blank lines, and compiler statements in front of a compilation unit. As a result, routines with UNIVAC FORTRAN V dependencies would not compile after being instrumented or restructured by FAVS. This was corrected by retaining all comments, blank lines, and compiler statements in front of FORTRAN V compilation units.

6.3 FORTRAN V INTERNAL SUBROUTINES

UNIVAC FORTRAN V allows subroutines and functions to be defined within one compilation unit. This is done by inserting each subroutine or function (minus an "END" statement) immediately in front of the "END" statement for the compilation unit. Such internal subroutines may be referenced only within the same compilation unit. Each internal sub-

routine has its own scope for labels; that is, the same label may be used in the main body of the compilation unit and one or more internal subroutines, and a label in an internal subroutine cannot be referenced from the main body of the compilation unit. This feature of the internal subroutines was not recognized during the initial FAVS installation. Instrumentation of compilation units with internal subroutines resulted in references to undefined labels when the instrumented code was compiled. This was corrected by creating an internal subroutine at the end of each instrumented compilation unit rather than referencing a label defined there. Similarly, when internal subroutines referenced duplicate labels, the restructured source would be incorrect. FAVS label processing has been corrected to allow duplicate labels within internal subroutines.

6.4 "EOF" VARIABLE

An early version of FORTRAN (RUN) on the CDC 6400 included "EOF" as a keyword in certain statement types. Recognition of this statement type was removed from FAVS, thus allowing the unrestricted use of "EOF" as a variable name in UNIVAC FORTRAN V.

6.5 ASSIGN -GO-TO STATEMENTS

UNIVAC FORTRAN V allows ASSIGN -GO-TO statements with an empty list of possible branch label destinations. This was identified as an error by FAVS and caused an incomplete graph to be used during instrumentation, restructuring, and static analysis. FAVS was modified to internally construct and use the complete list of possible branch label destinations for each ASSIGN -GO-TO statement.

6.6 LONG DATA STATEMENTS

Previously very large data statements caused an abnormal termination during FAVS processing. FAVS has been corrected to check for statements with more than 19 continuation lines, or executable statements with more than 250 symbols, keywords, operators, and delimiters.

These are treated as fatal errors which terminate FAVS processing after all source text has been scanned. An informative diagnostic identifies each statement which is too long.

APPENDIX A
UPDATES TO THE DMATRAN USER'S GUIDE

Appendix A consists entirely of updated pages for the January 1979 edition of the DMATRAN User's Guide, available as RADC-TR-78-268, Vol. I. Replacement by the modified pages in this appendix will update the DMATRAN User's Guide to indicate changes made during the FAVS Enhancement effort.

4 DMATRAN CONSTRAINTS

4.1 SYNTAX

- A maximum of 20 card images per statement.
- Statement labels between 10000 and 19999 should not be used because the DMATRAN preprocessor adds statement labels, beginning with label 19999 counting backwards, to the FORTRAN source code (Fig. 3.3).
- Don't transfer to labeled DMATRAN statements with FORTRAN GO TO's.
- Comments may not be interspersed within DMATRAN statements.
- All two-word DMATRAN directives may be written as two separate words or merged into one; i.e., DO UNTIL or DOUNTIL.

4.2 DO UNTIL

Remember, when the DO UNTIL...ENDUNTIL construct is used for iteration, the statements contained within the construct will be executed once before the logical expression is evaluated.

4.3 CASE

The value of <integer-expression> in CASE statements must be positive and must be less than 100.

4.4 BLOCK CONSTRUCT

- Each BLOCK...END BLOCK construct should occur after all INVOKE statements which refer to the block name, but may be before or after the RETURN statement.
- Blocks can only be entered through INVOKE statements. Sequential control transfers around BLOCK...END BLOCK constructs. Do not use a GO TO enter the middle of a BLOCK..END BLOCK construct from outside the block.

- The maximum number of INVOKEs and BLOCKs depends on the lengths of the BLOCK names and number of invocations, see Sec. 2.5.

RADC HONEYWELL 6180/MULTICS
SAMPLE DMATRAN JOB STREAM
(USING THE GCOS ENCAPSULATOR)

In order to use the DMATRAN precompiler, using source code written in DMATRAN generated by a programmer or by FAVS restructurer, the job stream shown in the following example can be used.

```
1.  $   snumb   (number)
2.  $   ident
3.  $   program rlhs
4.  $   limits  (CP time limit),32k,,(print line limit)
5.  $   prmfl   h*,r,r,>udd>3201c0320>Urban>dmatran>hstar
6.  $   select  >udd>3201c0320>Urban>dmatran>filedefs -ascii
7.  $   prmfl   01,r,s,>udd(BCD dmatran source file)
8.  $   prmfl   03,w,s,>udd>(BCD Translated FORTRAN source file)
9.  $   endjob
```

DMA UNIVAC 1100/80

SAMPLE DMATRAN JOB STREAM

The job stream in the following example can be used to execute the DMATRAN precompiler.

```
@ASG,A      YOURSOURCE.          .YOUR DMATRAN SOURCE
@USE        Y.,YOURSOURCE.       .
@ASG,A      DBM*FAVS-DMA.        .ASG DMATRAN PRECOMPILER
@USE        DMA.,DBM*FAVS-DMA.   .
@XQT        DMA.TRAN             .EXECUTE DMATRAN PRECOMPILER
@ADD        Y.ELEMENTS           .ADD DMATRAN SOURCE ELEMENTS HERE
```

The UNIVAC 1100/80 installation of the DMATRAN precompiler supports an additional command (see Sec. 5.1) to assist in compiling translated DMATRAN. This command contains CFOR or CFTN in columns 1 thru 4, followed by any desired information in columns 5 thru 80. The DMATRAN precompiler changes the C in column 1 of all CFOR and CFTN commands to an @ character as the command is written to the FORTRAN output file. When the DMATRAN precompiler automatically adds the FORTRAN output file to the runstream, the translated CFOR or CFTN statements request either the FORTRAN V or the ASCII FORTRAN compiler. Note that to compile a DMATRAN source element, the first line in the element should be a CFOR or CFTN command. Indented listings without FORTRAN V or ASCII FORTRAN compilations may be obtained by omitting CFOR and CFTN commands.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.